Pyreverse

Génération automatique de diagrammes UML

Par GALODE Alexandre

Date de publication : 1 novembre 2016

TOUT PUBLIC

Lorsque l'on développe un logiciel un tant soit peu important, en termes de code, on se trouve rapidement face au besoin de tracer les noms des classes, des variables…

Pour y parvenir, nous utilisons habituellement les diagrammes UML. Il peut cependant s'avérer pénible, selon la taille de notre projet de le faire à la main.

Bonne nouvelle, il existe en Python, un moyen d'automatiser tout cela. Partons à la découverte de Pyreverse.

Commentez

I - Introduction

Pyreverse est un outil en ligne de commande qui permet de scanner le code Python d'un projet afin de générer, de façon entièrement automatique, un diagramme UML.

Cet outil, développé par la société Logilab, est depuis septembre 2008 intégré à Pylint. Une simple installation via pip vous permettra alors d'en disposer pleinement.

Dans cet article, tous les codes sont exécutés en Python 3 exclusivement.

II - Possibilités offertes

Pyreverse permet de générer des diagrammes UML avec les caractéristiques suivantes :

- attributs de classes, avec si possible leur type ;
- méthodes de classe ;
- liens d'héritage entre classes ;
- liens d'association entre classes ;
- représentations des exceptions.

Côté fichier de sortie, Pyreverse ne propose nativement que deux formats : le *dot* et le *vcg*.

Même si le fichier généré ne correspond pas à 100 % à vos attentes, il vous permettra tout de même de vous simplifier un minimum le travail.

III - Ajoutons des formats de sortie

Comme vu précédemment, par défaut, seuls deux formats de sortie sont possibles : *dot* et *vcg*.

Cela limitant, basiquement, beaucoup les possibilités, il est possible d'augmenter la quantité des formats de sortie : l'installation de GraphViz.

Graphviz ne doit pas être installé depuis pip, mais depuis l'installeur dédié à votre OS, que vous pourrez trouver sur ***le site officiel*** *de l'outil.*
Une fois installé, il vous faudra modifier votre variable d'environnement PATH sous Windows, afin d'y ajouter le chemin vers le dossier « GraphvizX.YZ\bin ».

Pyreverse vous propose alors les formats suivants en sus :

canon	cmap	cmap_	dia	dot	eps	fig	gd	gd2	gif	hpgl	imap
imap_	ismap	jpe	jpeg	jpg	mif	mp	pcl	pdf	pic	plain	plain-ext
png	ps	ps2	svg	svgz	tk	vml	vmlz	vrml	vtx	wbmp	xdot
xlib											

Les formats proposés deviennent tout de suite plus appréciables, je pense notamment aux formats images (*png*, *jpg*), au *pdf*, ou encore le format *dia* pour faire de l'édition facilement.

Vous l'aurez compris, dans les faits, Pyreverse sera quasiment toujours utilisé avec Graphviz.

Pour l'ensemble des manipulations qui suivront, nous considérerons que vous avez correctement installé et paramétré Pyreverse et Graphviz.

IV - Les options

Pyreverse dispose d'un certain nombre d'options. Voici un récapitulatif des principales :

Option courte	Option verbeuse	Description
-a <level>	--show-ancestors=<level>	Permet de stipuler combien de niveaux d'héritage afficher.
-A	--all-ancestors	Permet d'afficher tout l'arbre d'héritage.
-b	--show-builtin	Permet d'afficher les classes des objets **builtin** dans le diagramme.
-f <mode>	--filter-mode=<mode>	Permet de filtrer ce qu'il faut faire apparaître dans le diagramme. Quatre filtres sont possibles : • PUB_ONLY ; • SPECIAL ; • OTHER ; • ALL. Par défaut, le filtre est à PUB_ONLY.
-k	--only-classnames	Génère un diagramme exclusivement avec les noms des classes.
-m [yn]	--module-names=[yn]	Permet d'inclure ou non ([yn] à y pour oui (*yes*) ou à n pour non (*no*)) le nom des modules dans la représentation des classes. Pas de valeur par défaut.
-o <format>	--output=<format>	Permet de stipuler le format de sortie. Le format doit être l'un de ceux stipulés précédemment. Par défaut, il s'agit du format *dot*.
-p <project name>	--project=<project name>	Permet de stipuler le nom du projet pour la génération du nom du fichier de sortie.
-s <level>	--show-associated=<level>	Permet d'afficher les liaisons liées aux imports (création d'objets) sur le nombre de niveaux stipulés.
-S	--all-associated	Permet d'afficher toutes les liaisons liées aux imports.

V - Mise en œuvre

Pour mettre en œuvre cet outil, nous allons nous appuyer sur un exemple déjà utilisé dans le **tutoriel sur l'héritage**. Nous aurons quatre classes :

- Voiture
- Citroen
- CB
- CitroenDs

La dernière héritera des deux premières et possédera des attributs privés et protégés. On y surchargera également la fonction spéciale « __del__ ». Enfin, nous considérerons que le propriétaire ajoute une CB dans sa voiture.

La commande est à exécuter dans le dossier où se trouve le code que l'on désire analyser.

Nous aurons donc la structure suivante :

voiture.py
```
1. class Voiture():
2.     def __init__(self):
3.         self.nombre_roues = 4
4.         self.nombre_fauteuils = 1
5.         self.moteur = False
6.         self.volant = True
7.
8.     def start_moteur(self):
9.         self.moteur = True
10.         return self.moteur
11.
12.     def stop_moteur(self):
13.         self.moteur = False
14.         return self.moteur
15.
16.     def statut_moteur(self):
17.         return self.moteur
18.
19.
20. if __name__ == "__main__":
21.     ma_voiture_basique = Voiture()
22.     print(ma_voiture_basique.statut_moteur())
23.     ma_voiture_basique.start_moteur()
24.     print(ma_voiture_basique.statut_moteur())
```

citroen.py
```
1. class Citroen():
2.     def __init__(self):
3.         self.type_suspension = "Hydractives"
4.         self.logo = "Chevrons"
5.         self.marque = "Citroen"
6.
7.
8. if __name__ == "__main__":
9.     ma_citroen = Citroen()
10.     print(ma_citroen.type_suspension)
11.     print(ma_citroen.logo)
```

cb.py
```
1. class CB():
2.     def __init__(self):
3.         self.marque = "Citizen-Band"
4.
5.
6. if __name__ == "__main__":
7.     ma_cb = CB()
```

cb.py
```
8.      print(ma_cb.marque)
```

citroen_ds.py
```
1. from voiture import Voiture
2. from citroen import Citroen
3. from cb import CB
4.
5.
6. class CitroenDs(Voiture, Citroen):
7.      def __init__(self):
8.          Voiture.__init__(self)
9.          Citroen.__init__(self)
10.          self.modele = "DS moderne (>2000)"
11.          self._option_payante_ds_01 = False
12.          self._option_payante_ds_02 = False
13.          self._option_payante_ds_03 = False
14.          self.__option_calculateur_ds_01 = False
15.          self.__option_calculateur_ds_02 = False
16.          self.__option_calculateur_ds_03 = False
17.          self.start_options()
18.          self.cb = CB()
19.
20.      def __del__(self):
21.          print("Suppression de la voiture")
22.
23.      def start_options(self):
24.          if _option_payante_ds_01:
25.              print("GPS activé")
26.          if _option_payante_ds_02:
27.              print("Anti dépassement lignes blanches activé")
28.          if _option_payante_ds_03:
29.              print("Freinage urgence activé")
30.
31.          if __option_calculateur_ds_01:
32.              print("Puissance moteur: 120 cv")
33.          elif __option_calculateur_ds_02:
34.              print("Puissance moteur: 150 cv")
35.          elif __option_calculateur_ds_03:
36.              print("Puissance moteur: 180 cv")
37.
38.
39. if __name__ == "__main__":
40.      ma_ds = CitroenDs()
41.      print(ma_ds.marque)
42.      print(ma_ds.modele)
43.      print(ma_ds.type_suspension)
44.      print(ma_ds.statut_moteur())
45.      ma_ds.start_moteur()
46.      print(ma_ds.statut_moteur())
```

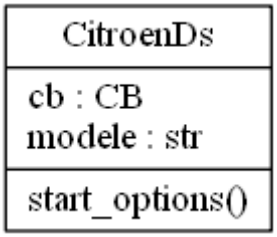
Afin de mieux appréhender le fonctionnement des principales options, je vous propose d'y aller progressivement.

V-A - Utilisation basique

Tout d'abord, aucune option précisée :

```
1. Pyreverse -o png citroen_DS.py
```

Nous récupérons un fichier nommé « classes_No_Name.png ». Ce diagramme représente la classe présente dans le module, avec ses attributs et méthodes.



Le « No_Name » du nom du fichier vient du fait que nous n'avons stipulé aucun nom de classe lors de l'appel de Pyreverse, chose que nous corrigerons à partir de maintenant.

V-B - Stipuler le nom du fichier généré

```
1. Pyreverse -o png -p demo_DVP citroen_DS.py
```

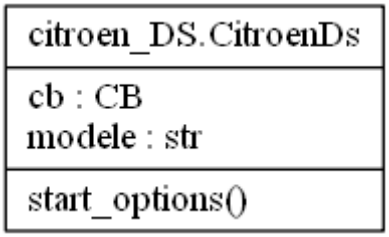
Le fichier généré porte désormais le nom de « classes_demo_DVP.png ».

V-C - Afficher les noms des modules

Remarquons tout de même que si les classes figurent bien, nous ne savons dans quel module elles se trouvent. Cela peut être mis à jour via l'option « -my » :

```
1. Pyreverse -o png -p demo_DVP -my citroen_DS.py
```

Le diagramme généré est désormais plus clair.



Le nom du module y figure désormais, rendant plus aisée la localisation du code.

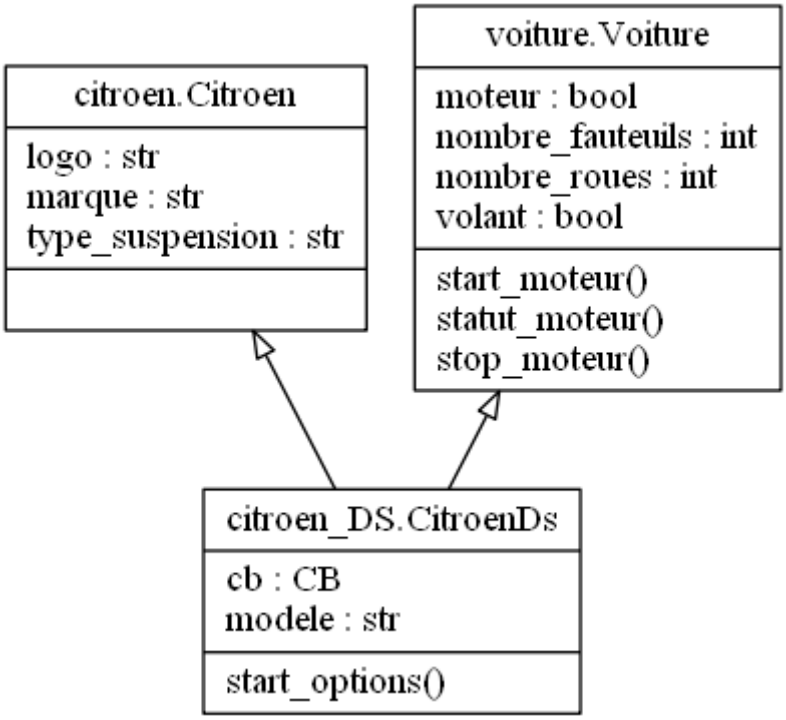
V-D - Afficher l'héritage

V-D-1 - Afficher tous les héritages

Voyons maintenant l'option « -A » :

```
1. Pyreverse -o png -p demo_DVP -A -my citroen_DS.py
```

Le nouveau diagramme montre clairement les divers héritages de notre classe CitroenDs.

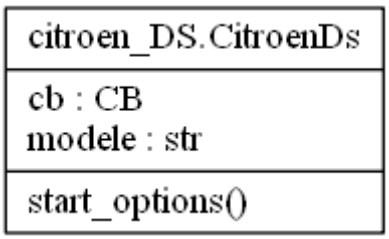


V-D-2 - Stipuler le niveau d'héritage à afficher

Si vous le désirez, vous pouvez stipuler expressément combien de niveaux d'héritage afficher, afin de vous concentrer sur une classe précise par exemple, ou encore alléger le diagramme.

Notre exemple ne contenant qu'un seul niveau, je vous propose de demander à n'avoir aucun héritage affiché en stipulant comme nombre de niveaux « 0 ».

```
1. Pyreverse -o png -p demo_DVP -a0 -my citroen_DS.py
```



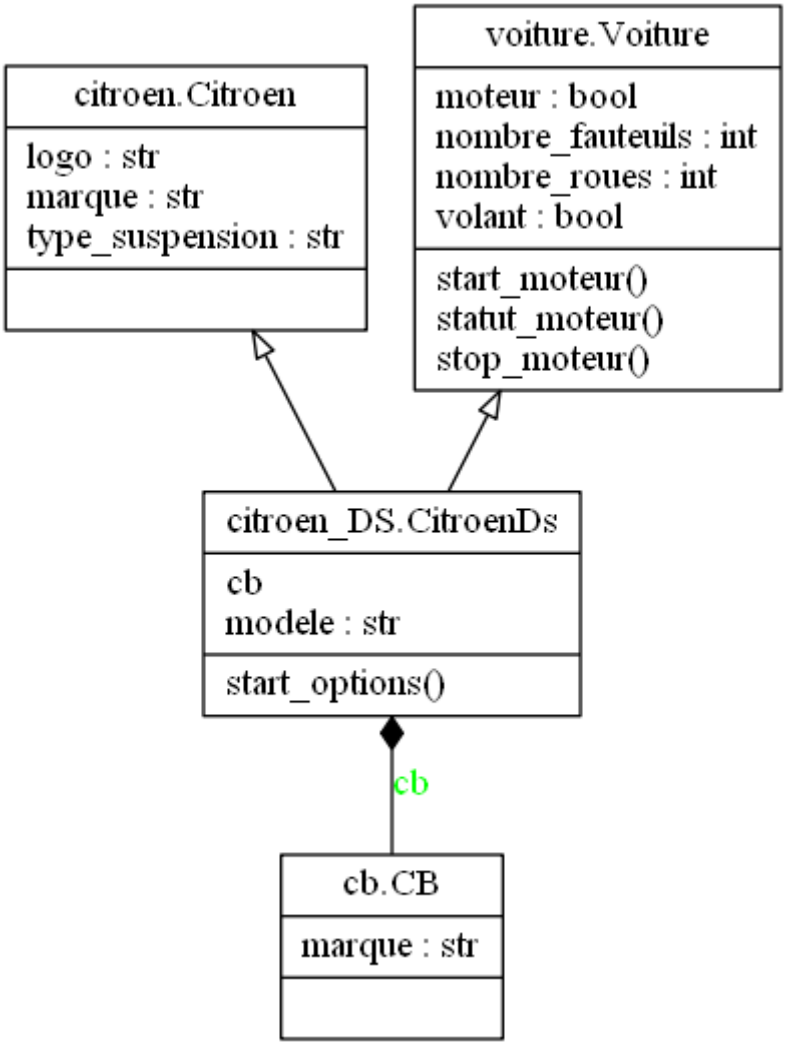
V-E - Afficher les imports et les objets

Quand vous générez le diagramme d'une classe, il peut être pratique de connaître les liaisons avec les divers imports.

V-E-1 - Afficher toutes les liaisons

Pour cela, nous allons utiliser l'option « -S ».

```
1. Pyreverse -o png -p demo_DVP -A -my -S citroen_DS.py
```



Dans ce diagramme, nous voyons clairement, que l'objet « cb » de notre classe CitroenDs dérive de la classe « cb.CB ». Au passage, remarquez en vert le nom de l'objet ; et que la classe dont provient « cb » figurant sur le diagramme, n'est plus stipulé au niveau de la classe « CitroenDs ».

V-E-2 - Afficher n niveaux de liaison

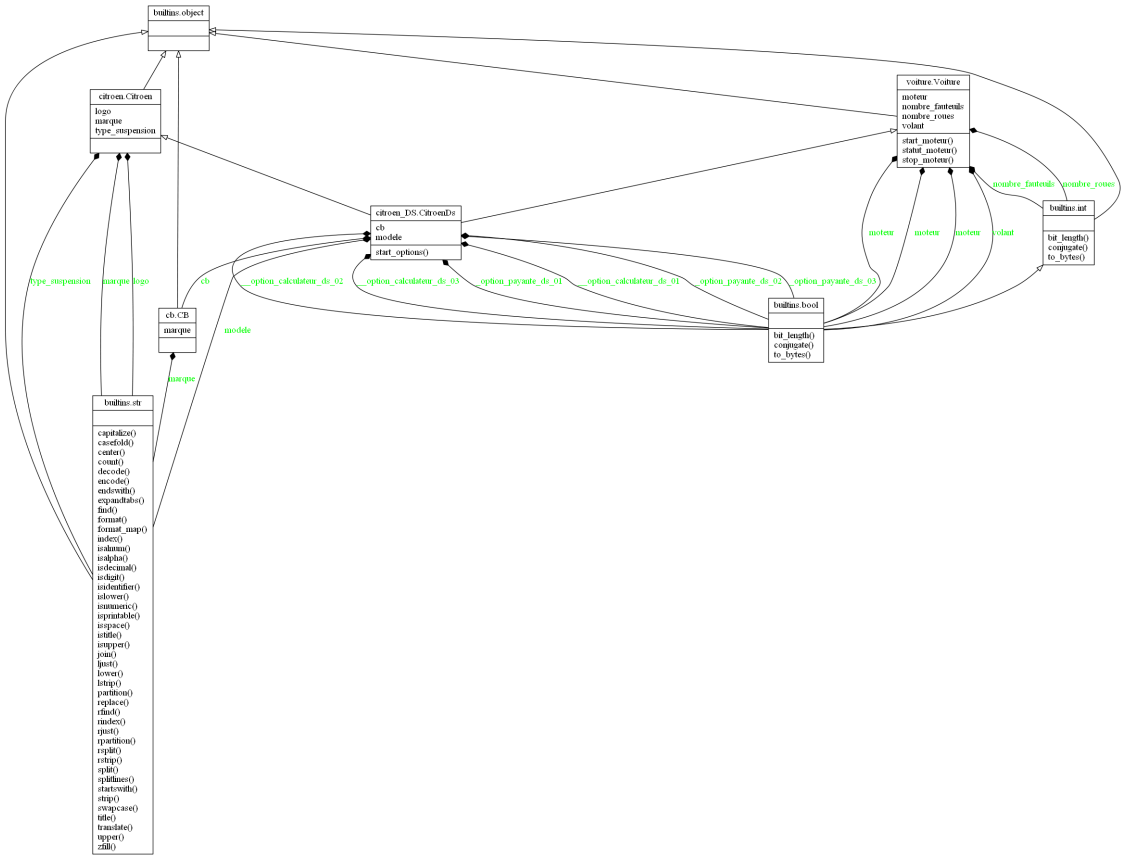
Nous allons utiliser ici l'option « -s ». Cette option fonctionne exactement comme l'option « -a ». Pour obtenir le diagramme précédent, il suffit de saisir la ligne de commande suivante.

```
1. Pyreverse -o png -p demo_DVP -A -my -s1 citroen_DS.py
```

V-E-3 - Afficher les objets builtins

S'il est possible d'afficher les imports et objets de notre code, on peut également vouloir faire apparaître ceux fournis par défaut avec Python, ce qu'on appelle les *builtins*. Pour cela, il faut utiliser l'option « -b ». Et en Python, tout étant objet, votre diagramme va rapidement prendre un peu d'espace.

```
1. Pyreverse -o png -p demo_DVP -A -my -S -b citroen_DS.py
```



V-F - Filtrer les informations

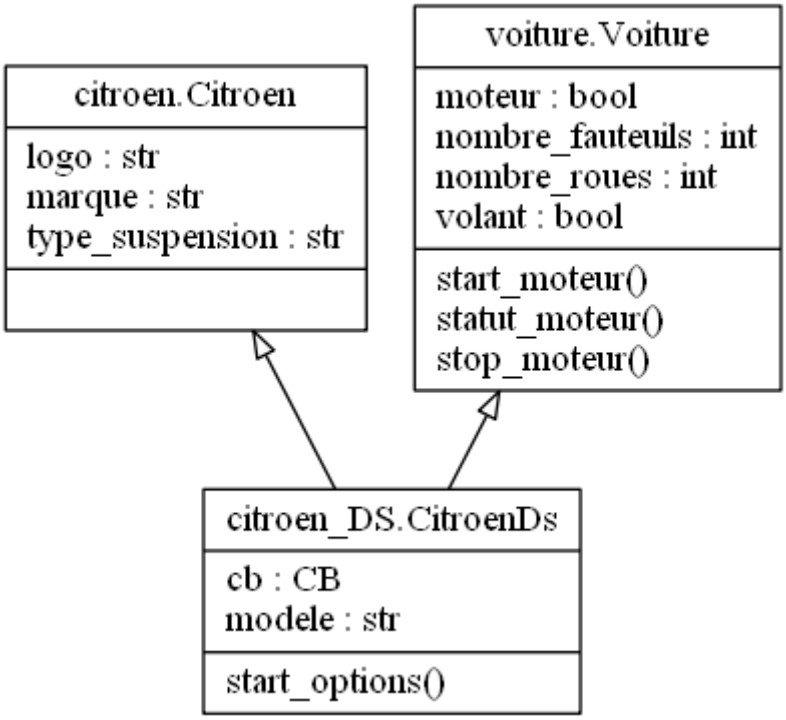
Vous disposez de la possibilité de filtrer les méthodes et attributs affichés sur le diagramme, via l'option « -f ». Quatre possibilités vous sont offertes, vous permettant de coller au plus près de vos besoins.

V-F-1 - Filtre PUB_ONLY

Il s'agit de l'option « -f PUB_ONLY ».

```
1. Pyreverse -o png -p demo_DVP -A -my -f PUB_ONLY citroen_DS.py
```

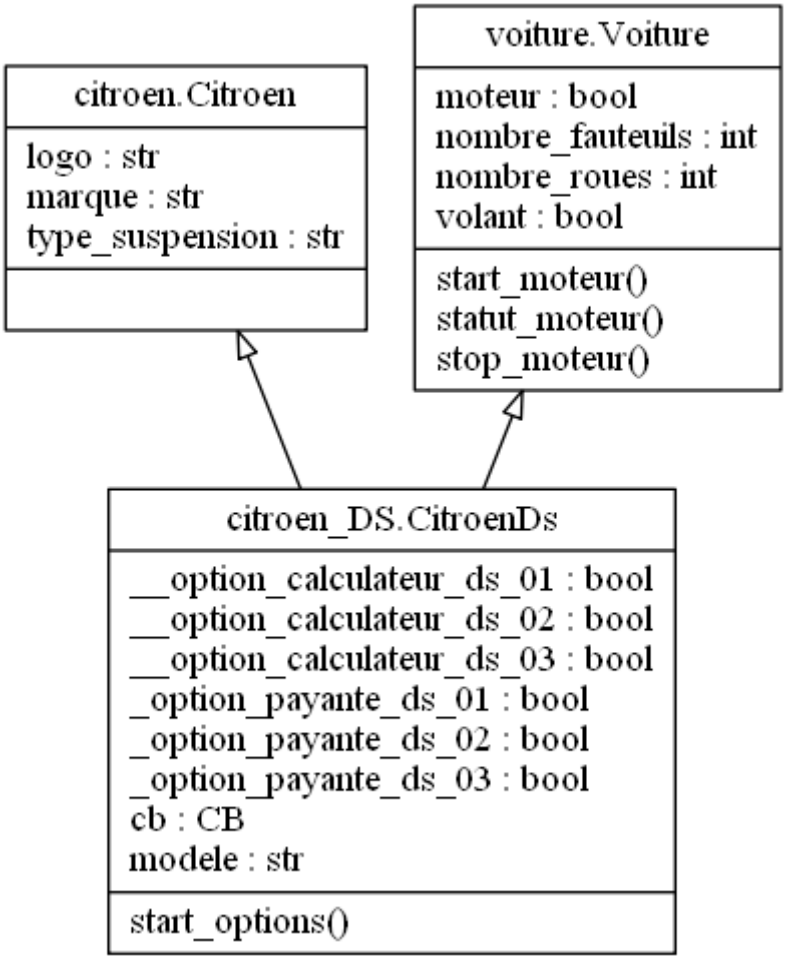
Vous ne voyez aucune différence avec la commande sans l'option « -f ». Normal, il s'agit de l'option par défaut.



V-F-2 - Filtre SPECIAL

Le filtre « -f SPECIAL » filtre les méthodes spéciales (__init__, __del__…). En revanche, aucun filtrage n'est effectué sur les attributs.

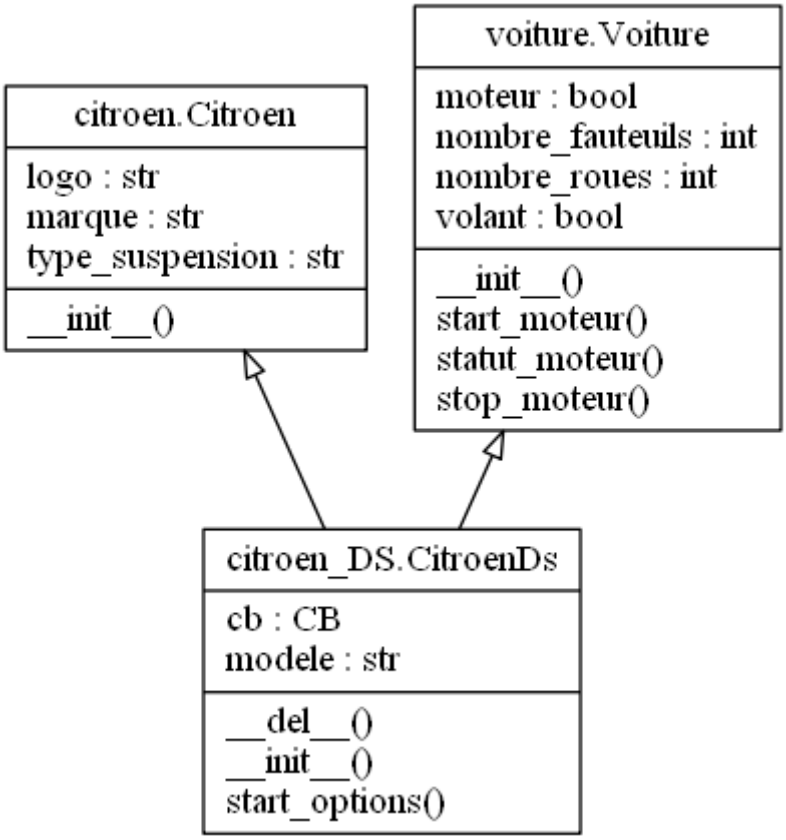
```
1. Pyreverse -o png -p demo_DVP -A -my -f SPECIAL citroen_DS.py
```



V-F-3 - Filtre OTHER

L'option « -f OTHER », est l'inverse du filtre SPECIAL. Il vous permet de ne pas afficher les attributs protégés et privés, mais d'afficher tout de même, toutes les méthodes, quelles qu'elles soient.

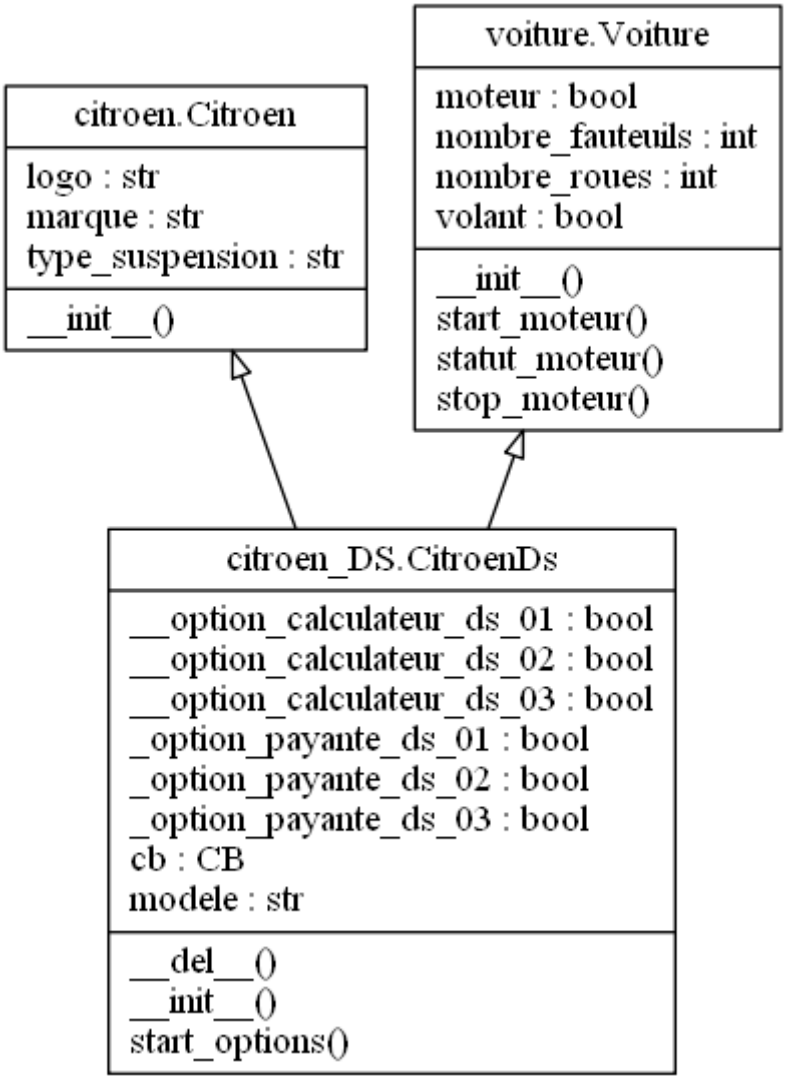
```
1. Pyreverse -o png -p demo_DVP -A -my -f OTHER citroen_DS.py
```



V-F-4 - Filtre ALL

Si vous désirez tout afficher sans distinction, ni filtrage précis, il vous suffit d'utiliser « -f ALL ».

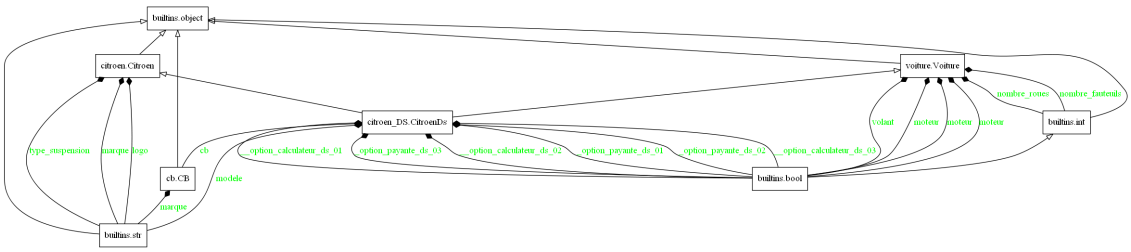
```
1. Pyreverse -o png -p demo_DVP -A -my -f ALL citroen_DS.py
```



V-G - Épurer le diagramme

Vient enfin la dernière option, « -k ». Cette dernière vous permet d'épurer l'affichage en le limitant simplement aux modules, classes et objets. Comprenez que nous ne ferons alors figurer ni attributs, ni méthodes.

```
1. Pyreverse -o png -p demo_DVP -A -my -S -b -k citroen_DS.py
```



VI - Conclusion

Comme nous venons de le voir, l'outil Pyreverse est très simple d'emploi.

Couplé avec Graphviz, il vous permettra de générer simplement des diagrammes UML, éditables ou non selon le format utilisé, qui vous seront fort utiles.

J'espère que cet article vous a plu et que la découverte de cet outil vous permettra de vous simplifier la vie pour vos futurs développements.

VII - Remerciements

Merci aux personnes suivantes pour leur aide :

- **f-leb**
- **tyrtamos**
- **Pascaltech**